PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 3/00 — A1

(11) International Publication Number: WO 00/58817

(43) International Publication Date: 5 October 2000 (05.10.00)

(21) International Application Number: PCT/US00/08325

(22) International Filing Date: 29 March 2000 (29.03.00)

(30) Priority Data:
09/281,737 30 March 1999 (30.03.99) US

(71) Applicant: FLASHPOINT TECHNOLOGY, INC. [US/US]; 152 N. First Street, No. 800, San Jose, CA 95112 (US).

(72) Inventor: PAVLEY, John, F.; 10291 Norwich Avenue, Cupertino, CA 95014 (US).

(74) Agents: SULLIVAN, Stephen, G. et al.; Sawyer Law Group LLP, P.O. Box 51418, Palo Alto, CA 94303 (US).

(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

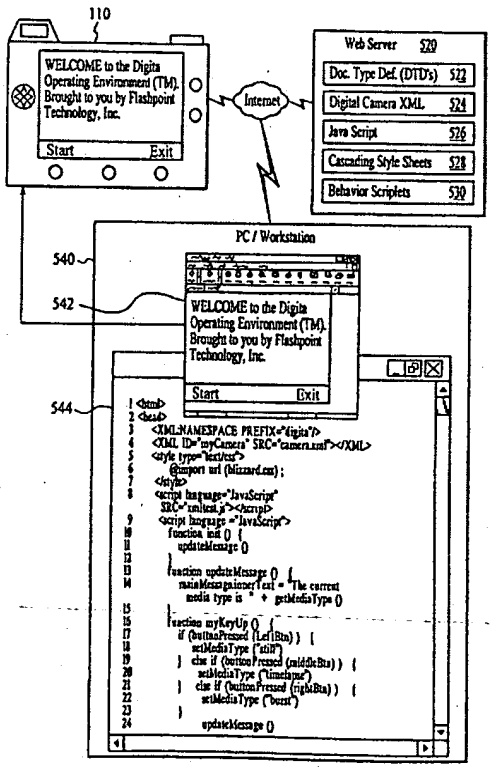
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND SYSTEM FOR PROVIDING A DIGITAL IMAGING DEVICE WITH A WEB-BASED GRAPHICAL-USER-INTERFACE

(57) Abstract

A method and system for providing a digital imaging device (110) having a display (402) with a Web-based graphical user interface (GUI) that is updateable via downloads from the Web. The method and system comprise providing the digital imaging device with a meta-language application (500) defining the GUI, and a meta-language parser (501) for interpreting and displaying the meta-language application on the display. The method and system further include updating the meta-language application and posting the updated meta-language application on a remote server for distribution; and downloading the updated meta-language application into the digital imaging device from the remote server (520) to thereby change the GUI of the camera.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND SYSTEM FOR PROVIDING A DIGITAL IMAGING DEVICE WITH A WEB-BASED GRAPHICAL-USER-INTERFACE

FIELD OF THE INVENTION

The present invention relates to a method and system for system for providing a digital imaging device with a Web-based graphical user interface (GUI).

BACKGROUND OF THE INVENTION

The use of digital imaging devices, such as digital cameras, is becoming widespread. Today's digital cameras look and function similar to analog 35mm SLR's cameras, except that digital cameras store images digitally and also include an LCD display that allows the user to look at the images directly on the camera.

FIG. 1 is a diagram illustrating a back and top view of a conventional digital camera. The camera 110 includes an LCD screen 402, a four-way navigation control button 409, an overlay button 412, a menu button 414, and a set of programmable soft keys 416. FIG. 2B is a top view of the camera 110 showing a shutter button 418, and a mode dial 420. The camera may optionally include status LCD 406, status LCD scroll and select buttons 422 and 424, a sound record button 426, and zoom-in, zoom-out buttons 426a and 426b.

Typically, all features and functions of the digital camera can be accessed and operated via the LCD screen 402, where the operation of the camera is provided by some type of mode-driven user interface. The assignee of the present invention has developed an operating system referred to as Digita™ for controlling the operation of a digital camera through a menu-driven graphical-user-interface (GUI).

A digital camera equipped with such an operating system supports four operating modes; capture (record) mode, review mode, play mode, and connect mode. The user typically switches between the capture, review, play, and connect modes using the mode dial 420. When the camera is placed into a particular mode, that mode's default GUI screen appears in the LCD screen 402 in which a set of items are displayed, such as images, icons, and text.

In capture mode, the camera 110 is capable of capturing images through the use of either the LCD screen 402 or with an optical viewfinder (not shown). Typically, the camera may capture both still images and sequential images, such as time-lapse and burst images.

Review mode, which is shown in FIG. 1, allows the user to review camera contents and sort the images. In play mode, the camera 110 allows the user to view screen-sized images in the LCD screen. Play mode also allows the user to hear recorded sound associated to a displayed image, and to play back sequential images. And connect mode allows the user to print and transfer their images.

Most of the GUI that controls the operation and look and feel of the camera is implemented as a hard-coded software application that is written in a high-level programming language, such as C++. The implementation of this type of software application requires a lengthy development process.

The first stage of the development process is to design the software, which entails defining the software requirements and developing a detailed design description. After the design has been completed, the program is written in the appropriate programming language to provide source code. After the source code has been generated, the source code is compiled into object code, and the object code is then tested and debugged in accordance with a test plan to make sure that the software performs as designed. After the functional performance of the software has been validated through many rounds of testing and debugging, the object code is loaded into the digital cameras, and the digital cameras are then shipped to the consumers.

Although equipping digital cameras with such a GUI enhances the camera's ease of use, the current method of implementing the GUI has its disadvantages. One disadvantage is that the GUI included in digital cameras is sold to a disparate group of consumers. For example, the same model camera may be purchased by novice users, expert users, and by vertical markets for use in the real estate and insurance industries, for instance. What may be a sufficient GUI application for one group of users may be insufficient for another group.

One possible solution is for the GUI developer to develop several different versions of the GUI for incorporation into the same camera model to provide the cameras with a different look and feel that is tailored to particular users. Another alternative is for the GUI developer to write new GUI applications and offer them to the camera owners as after-market upgrades. But since the GUI application software is hard-coded to the camera hardware, such a solution is untenable because of the difficulty and cost associated with development and cost of several different software versions. In addition, the GUI application would have to be compiled for each targeted hardware platform, which further complicates the development and maintenance process.

Accordingly, what is needed is a system and method for changing the look and feel of a digital camera's GUI without the cost and difficulty associated with having to rewrite and ship conventional types of GUI software applications. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides a method and system for providing a digital imaging device having a display with a Web-based graphical user interface (GUI) that is updateable via downloads from the Web. The method and system comprise providing the digital imaging device with a meta-language application defining the GUI, and a meta-language parser for interpreting and displaying the meta-language application on the display. The method and system further include updating the meta-language application and posting the updated meta-language application on a remote server for distribution; and downloading the updated meta-language application into the digital imaging device from the remote server to thereby change the GUI of the camera.

According to the system and method disclosed herein, implementing the GUI using a meta-language allows the GUI to be cross-platform compatible so that there is no need to compile the GUI application for different hardware platforms. And since the camera GUI is built using an industry standard document object model, no custom or proprietary PC development tools are required, which greatly reduces costs, maintenance, and time to market issues.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram illustrating a back and top view of a conventional digital camera.

FIG. 2 is a block diagram of one preferred embodiment of a digital camera 110 is shown for use in accordance with the present invention.

FIG. 3 is a flow chart illustrating the process of providing a digital camera with a cross-platform Web-based graphical user interface.

FIG. 4 is a block diagram illustrating the software organization in the non-volatile memory in which the digital camera software is stored.

FIG. 5 is a diagram illustrating the XML-related files that are posted on a Web server to allow for updating the camera GUI via Internet downloads.

FIGS. 6A and 6B are diagrams illustrating the changing of an example camera

GUI screen in accordance with the present invention.

FIG. 7A is a partial listing of a CSS file that defines the appearance for a soft key label shown in FIG. 6A.

FIG. 7B is a partial listing of an updated CSS file that defines the appearance for a soft key label shown in FIG. 6B.

FIG. 8 is a listing of an example HTML file with embedded XML data for generating the text and soft keys labels for the GUI screens of FIGS. 6A and 6B.

FIG. 9 is a partial listing of an example digital camera XML file.

FIG. 10 is a partial listing an example DTD file.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to a method and system for providing a digital imaging device with a Web-based graphical user interface. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Although the present invention will be described in the context of a still digital camera, various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. That is, any digital imaging capture device which captures, stores, or displays digital images, could incorporate the features described hereinbelow and that device would be within the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

Referring now to FIG. 2, a block diagram of one preferred embodiment of a digital camera 110 is shown for use in accordance with the present invention. Camera 110 preferably comprises an imaging device 114, a system bus 116 and a computer 118. Imaging device 114 includes an image sensor, such as a charged coupled device (CCD) or a CMOS sensor, for generating a set of raw image data representing a captured image. In a preferred embodiment, system bus 116 provides connection paths between imaging device 114, an optional power manager 342, central processing unit (CPU) 344, dynamic random-access memory (DRAM) 346, input/output interface (I/O) 348, non-volatile memory 350, and buffers/connector 352 that connect an optional removable memory 354 to system bus 116.

CPU 344 may include a conventional microprocessor device for controlling the

operation of camera 110. In the preferred embodiment, CPU 344 is capable of concurrently running multiple software routines to control the various processes of camera 110 within a multithreaded environment. For example, images may be captured at the same time that previously captured images are processed in the background to effectively increase the capture rate of the camera. In a preferred embodiment, CPU 244 runs an operating system that includes a menu-driven GUI and provides image processing through software, rather than hardware, such as the Digita™ Operating Environment. Although CPU 344 is preferably a microprocessor, one or more DSP's (digital signal processor) or ASIC's (Application Specific Integrated Circuit) could also be used.

I/O 348 is an interface device allowing communications to and from computer 118. For example, I/O 348 permits an external host computer (not shown) to connect to and communicate with computer 118. I/O 348 also interfaces with a plurality of buttons and/or dials 404, and an optional status LCD 406, which in addition to the LCD screen 402, are the hardware elements of the camera's user interface 408.

Non-volatile memory 350, which may typically comprise a conventional read-only memory or flash memory, stores a set of computer-readable program instructions to control the operation of camera 110. Removable memory 354 serves as an additional image data storage area and is preferably a non-volatile device, such a flash disk, readily removable and replaceable by a camera 110 user via buffers/connector 352.

Power supply 356 supplies operating power to the various components of camera 110. Power manager 342 communicates via line 366 with power supply 356 and coordinates power management operations for camera 110. In the preferred embodiment, power supply 356 provides operating power to a main power bus 362 and also to a secondary power bus 364. The main power bus 362 provides power to imaging device 114, I/O 348, non-volatile memory 350 and removable memory 354. The secondary power bus 364 provides power to power manager 342, CPU 344 and DRAM 346. Power supply 356 is connected to main batteries 358 and also to backup batteries 360. Power supply 356 may also be connected to an external power source.

Dynamic Random-Access-Memory (DRAM) 346 is a contiguous block of dynamic memory that may be selectively allocated for various storage functions. DRAM 346 stores both raw and compressed image data and is also used by CPU 344 while executing the software routines used within computer 118. The raw image data received from imaging device 114 is temporarily stored in several input buffers (not shown) within DRAM 346. Once the raw image data is processed, it is stored in a frame buffer (not shown) for

display on the LCD screen 402. After processed image data has been stored in DRAM 346, LCD controller 390 transfers the image data to LCD screen 402 for display.

As stated previously, the user operates and accesses the functions of the camera 110 via a GUI, as shown in FIG. 1. The present invention is a method and system for providing a digital imaging device with a Web-based graphical user interface (GUI). Rather than writing the camera GUI in a compiled, high-level language that is difficult and expensive to change, such as C++, the present invention implements the camera GUI using a meta-language language so that the look and feel of the GUI may be easily customized and changed via a network, such as the Internet.

Before describing the present invention in detail, a brief overview of meta-language's will be provided. A meta-language represents a machine-to-machine document format that is typically used on the World Wide Web. Common examples include HTML (Hypertext Markup Language) and XML (Extensible Markup Language). Both HTML and XML are derived from SGML, the Standard Generalized Markup Language, which is widely used to publish structured documents

Typical Web pages are documents that include HTML tags, or codes, embedded in the text. HTML defines the Web page's layout, fonts and graphic elements as well as the hypertext links to other documents on the Web. When a client computer accesses a Web page written in HTML, each page element is sent one file at a time, from the Web server to a Web browser on the client computer. The Web browser includes a meta-language parser that is capable of interpreting the HTML tags in order to display the defined page layout, fonts and other graphical elements. HTML tags tend to change with each new version of HTML, but are inflexible. Although programming-like statements are sometimes added to HTML, it is not considered a full-blown programming language such as Java or JavaScript. Rather it could be considered a "presentation language" because HTML tags describe the formatting (presentation), but not the meaning or behavior of objects within a document. Therefore, in a preferred embodiment, the Web-based camera GUI of the present invention is implemented using XML.

XML serves much of the same purposes as HTML, and the syntax of XML is very similar to that of HTML since HTML and XML both derive from SGML. XML, however, is a document description language that is more flexible than HTML. While HTML uses only predefined tags to describe elements within a document, XML allows tags to be defined by the developer of the document in order to represent meta-data,

which is information describing the content of the document. Thus, tags for virtually any data items can be used for specific applications. A Document Type Definition (DTD) defines these tags and their properties, including their relationships. The DTD can be separate from the XML document itself so multiple authors working on the same application can use it.

An XML DTD defines tags, but does not specify presentation. Presentation of XML documents depends on style sheets, which define the visual attributes of each tag type. Separating the presentation from the content allows much more flexibility. For example, by changing style sheets the same information contained in a tag can be displayed on a standard screen, displayed in large fonts, or formatted for printing. According to the present invention, this Web-based XML paradigm is extended to define a digital camera GUI to provide an easy an efficient way to modify the camera's GUI, as explained below.

Referring now to FIG. 3, a flow chart illustrating the process of providing a digital camera with a cross-platform, Web-based graphical user interface (GUI). The process begins by providing the digital camera with a meta-language application defining the camera's GUI in step 480. The camera is also provided with a meta-language parser for interpreting the meta-language application so that the GUI may be displayed on a display in step 482.

Subsequently, the meta-language application is updated and/or customized versions of the application are provided in step 484. The updated and/or customized versions of the meta-language application are then posted on a remote server for distribution over a network, such as the Internet, in step 486. Users may then access the remote server and download the updated meta-language application into the digital camera to thereby change the GUI of the camera in step 488.

In a preferred embodiment, the user may access the remote server by connecting the camera directly to the Internet using File Transfer Protocol (FTP) or HyperText Transport Protocol (HTTP) software. However, since the files comprising the GUI are Web-based and therefore cross-platform, a user may access the files using any device that can interpret XML files, such as a PC/workstation Web-browser, a palmtop Web-browser, or an XML-enabled Web-TV, for instance. And assuming that each device used the same XML application, the user could view the files on each device using the same consistent user interface.

FIG. 4 is a block diagram illustrating the software organization in the non-volatile

memory 350 in which the digital camera software is stored. The software 490 comprises the XML application 500, a tool box 504, a set of drivers 506, a kernel 508, and a startup/configuration module 510.

The XML application 500 is the program that controls the GUI and high-level functions of the digital camera and interfaces with functions in the tool box 504. The XML parser 501 interprets the XML application and displays the results on the LCD screen 402 or on an external display to provide the GUI, such as the one shown in FIG. 1. The tool box 504, in turn, controls how the digital camera captures and manipulates images. The drivers 506 help control the I/O 348 interface for external communication; the kernel 508 controls basic operating system functions; and the startup/configuration module 510 controls the camera's boot-up process.

When the camera 110 is first turned on and booted up, the startup/configuration 510 module begins to execute and loads the drivers 506 and the kernel 508 into DRAM 346. After the kernel 508 is loaded, the startup/configuration module loads the XML application 500 and XML parser 501 into DRAM 346, and control of the camera 110 is passed to the XML application 500 through the XML parser 501. In addition to interpreting the XML application to provide the GUI on the LCD screen 402, XML parser 501 could also perform other functions similar to that of a Web browser, such as launching plug-in's and running Java applets, for instance.

The camera 110 communicates with a network through protocol stack 502. The protocol stack 502 represents any network protocol the supports a persistent network connection, such as TCP/IP (Transmission Control Protocol/Internet Protocol), Point-to-Point Protocol, NetBIOS, IPX, and LU6.2, and link layer protocols, such as Ethernet, token ring, and ATM.

Protocol stack 502 interfaces between the XML application 500 and communications hardware such a modem (not shown), to provide a network connection with a remote server. When, for example, the camera 110 is connected to a remote Web server, HTML and/or XML files are received by protocol stack through the modem and transferred to the meta-language parser 501 for interpretation and display.

FIG. 5 is a diagram illustrating the XML-related files that are posted on a remote server to allow for changing the camera GUI via Internet downloads. In a preferred embodiment, the files are posted on a web server 520. The XML-related files posted on a web server 520 include one or more Document Type Definition (DTD) files 522, a digital camera XML file 524, a Java script 526 file, and several cascading style sheets 528 and

behavior scriptlet files 530. In one preferred embodiment, these same files except the DTD files 522 are also stored on the digital camera 110 to comprise the XML application 500 of FIG. 4. As shown, the files may be downloaded from web server 520 over the Internet directly into the camera 110, or download to a personal computer (PC) or workstation, and then downloaded into the camera 110.

The DTD files 522 are a family of public XML schemas that define the XML Application 500. The DTD's 522 model the runtime environment of the digital camera by defining the rules for implementing a valid GUI. The rules are in the form of XML tags defining the existence of elements comprising the GUI, such as soft keys, thumbnail images, icons, and so on.

The digital camera XML file 524 is based on the DTD files 522 and contains a specific instance or model of a camera as defined by the DTD's 522. In order to bring this model to life, behaviors are defined for the elements comprising the GUI. This is accomplished through helper scripts, called Java scripts 526, and behavior scriplets 530.

The behavior scriptlets 530 define a set of rules that implement the interaction or behavior of the GUI, such as defining for example, that when a soft key is depressed, the soft key label blinks, or for a kid's version of the GUI, a sound is played.

The cascading style sheets 528 (CSS) control appearance, rather than behavior, by defining the visual attributes of each tag type in the DTDs 522. Using the CSS's 528, graphical screen elements of the GUI are given color, size, font style, and position. Besides being displayed, the graphical screen elements of the GUI can also be or hidden as required by the GUI's design using the XSL, which maps XML to HTML and CSS's at runtime.

The behavior scriptlet files 530 together with CSS files 528 define a particular appearance or theme for the GUI. As long as the camera model defined by the digital camera XML file 524 obeys the DTD's 522, different behavior scriptlet files 530 and CSS files 528 can be swapped into the camera 110 from the Web server 520 to change the appearance of the GUI. For example, the original GUI shipped with the camera could be transformed into a "Football" or "Science Fiction" theme.

For a more radical change to the GUI, updates and/or new versions of the digital camera XML file 524 can be posted on the Web server 520, downloaded into the camera 110, and automatically installed to provide totally new GUIs. These new GUIs would have totally different behaviors and user interactions and could be targeted at specific customer or solution segments: Kids, Teens, Soccer Moms, Photographers, and so on.

In the embodiment where the files are downloaded via the PC, the user may tryout new GUI's by downloading different combinations of the XML-related files posted on the Web server 520, and interacting with the resulting GUI through the PC's Web browser 542.

In another aspect of the present invention, camera manufacturers and/or third party software developers may access the Web server 520 from their desktop and access the XML-related files on Web server 520 to author their own screens to provide new and customized versions of the GUI. This is done by editing the XML files using a text editor 544, and then testing the designs with the web browser 542. These new GUI's could then be posted on the developer's Web site, or provided to camera manufactures for incorporation into new digital cameras.

For example, a developer might write a custom file, called "strobe changer" for instance, that changes the state of the camera's flash by describing a screen that shows the current flash setting, and allows the user to change the setting by pressing the soft keys. The file might also define that when a user presses a soft key to change the setting, the soft key flashes to indicate the change. After creating this file, the developer would post the file on their Web site for users to download. The user would then download the file to the camera. When the file is run, the links specified in the file are used to access the corresponding digital camera XML file 524 stored in the camera and cause the camera's GUI to change accordingly.

Referring now to FIGS. 6A and 6B, diagrams illustrating the changing of an example camera GUI screen are shown in accordance with the present invention. The purpose of the GUI screen, which is displayed on the camera's LCD screen 402, is to inform the user what media type the camera is set to capture and to allow the user to change that setting. In the example shown, the camera is currently set to capture still images. The user may change the media type by pressing one of the soft keys 416 under the corresponding soft key label, which then updates the display accordingly.

FIG. 6A represents the screen as originally provided with the digital camera 110, while FIG. 6B represents an update to that GUI screen via file downloads from a remote server. In one embodiment of the present invention, the GUI screen may be changed by downloading updated cascading style sheets 528 (CSS's 528) 528, shown in FIG. 5.

FIGS. 7-10 are example program listings of the meta-language files comprising the camera's XML application 500 of FIG. 4 and the files posted on the Web server 520 of FIG. 5 for generating the GUI screens of FIGS. 6A and 6B.

Referring now to FIG. 7A, a partial listing of a CSS file is shown that defines the appearance of the soft key label for soft key sk1 shown in FIG. 6A. As shown, the CSS defines the soft key's position on the screen, width and height, color, font, and so on.

Referring now to FIG. 7B, a partial listing of an updated CSS file is shown that defines the appearance of the soft key label for soft keyto sk1 in the customized GUI of FIG. 6B. As in FIG. 7A, the CSS file sets forth the same position width and height, but the soft key label is given a different color, font, and border style. Thus, by providing minor updates to the CSS files, the camera GUI may be given widely varying appearances.

Referring now to FIG. 8, a listing of an example HTML file with embedded XML data for generating the text and soft keys labels for the GUI screens of FIGS. 6A and 6B is shown.

Line 3 of the file defines an XML name space, which is defined here as "Digita". In line 4, the file is linked to a camera model that is defined an XML-enabled camera application file called "camera.XML," and is given an ID of "mycamera." In line 5, the file is linked to a cascading style sheet, "text/css". In line 6, the file is linked to a behavior defined by a Javascript file called "XMLtest.js." Note here that the XML file, the style sheet, and the Javascript reside locally with the HTML file, however the references in the HTML file to the XML file in the style sheet could be provided as a link to the Web server.

Line 8 of the file defines a function called init ( ), which is invoked when the GUI screen is initialized and calls the function updatemessage ( ) in line 9. The function updatemessage( ) displays the text "The current media type is " on the display, and then calls the function getmediatype( ) to retrieve and display the current media type setting in lines 11 and 12.

Next, the file defines a function called mykeyup( ) in line 14 to control what occurs on the display when a soft key button 416 is pressed. If the left button is pressed then the media type is set to still, if the middle button is pressed then the media type is set to time lapse, and if the right button is pressed the media type is set to burst. After the media type has been selected by a button press event, the function updatemessage ( ) is called again in line 21 to update the display.

Referring now to FIG. 9, a partial listing of an example digital camera XML file 524 is shown, which defines a specific camera model from the DTD files 522. This digital camera XML file 524 is written HTML and the portion shown defines the media types that the camera is capable of capturing. The term "digita" in line 2 refers to the source DTD file 522, called "digita.DTD", and the terms "screen", "appName",

"scrName", "page", "cmdbar", and "sk1" through "sk3" (soft keys), are all tags defined in the DTD 522.

When creating an alternative camera model, the tags appearing in the file just need to be assigned an ID and text. In line 6, for example, the string mainmessage( ) is defined, where the actual text for the message is to be inserted between the angle brackets.

This particular XML model defines that the media types that may be captured are still, time lapse, and burst. Lines 8-12 define that these three media type selections are displayed as the three soft key labels (sk1-sk3) in the command bar (cmdbar), as shown in FIGS. 6A and 6B. The ID for soft key one (sk1) is set to the left button and is assigned the text "still", the ID for soft key two (sk2) is set to the middle button and is assigned the text "time lapse", and the ID for soft key three (sk3) is set to the right to button and assigned the text "burst".

Although only one XML file is shown here, those skilled in the art will appreciate that an actual camera GUI could include separate XML files to implement the features defined in the DTD files. For example, a camera that includes a play mode, a capture mode, a review mode, and a connect mode could be implemented using the following XML files: play.XML, capture.XML, review.XML, and connect.XML. Other XML files could include an image tags.XML and product parameters.XML files. All of these separate XML files would then comprise the digital camera XML file 524.

Referring now to FIG. 10, a partial listing an example DTD file is shown. Line 1 defines that the camera includes three parameters, or tags, relating to images: a media type, an image count, and a quality. Lines 3 through 5 define to that the three tags are parsed character data. Line 7 defines the attributes for the camera. In the example shown, the camera mode is the only attribute defined for the camera, but typically, there would be many more. The camera mode is defined as having four states: capture, review, play, and connect mode, and the default mode is set to capture mode. Lines 9 through 13 define entities for the camera GUI, where the entities in the example define the following characters: less than, greater than, ampersand, apostrophe, and quote.

In accordance with the present invention, the GUI of the digital camera 110 may be changed by downloading to the camera 110 any combination of the meta-language files described herein, such as cascading style sheets 528, digital camera XML files 524, and DTD's 522.

One primary advantage of implementing the GUI using a meta-language is it

makes the GUI cross-platform compatible, meaning that there is no need to compile the GUI application for separate hardware platforms. And since the camera XML-based GUI is built using an industry standard document object model, no custom or proprietary PC development tools are required, which greatly reduces costs, maintenance, and time to market issues.

In summary, a method and system for providing a digital imaging device with a Web-based graphical user interface has been disclosed. Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the present invention. For example, the present invention may be implemented in other types of digital imaging devices, such as an electronic device for archiving images that displays stored images on a television, for instance. In addition, software written according to the present invention may be stored on a computer-readable medium, such as a removable memory, and loaded into the digital camera for execution. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

CLAIMS

What is claimed is:

1 A method for providing a digital imaging device having a display with a Web-based graphical user interface (GUI), comprising the steps of:

a) providing the digital imaging device with a meta-language application defining the GUI;

b) providing the digital imaging device with a meta-language parser for interpreting and displaying the meta-language application on the display;

c) updating the meta-language application and posting the updated meta-language application on a remote server for distribution; and

d) downloading the updated meta-language application into the digital imaging device from the remote server to thereby change the GUI of the camera.

2 A method as in claim 1, wherein step a) further includes the steps of:

i) defining the meta-language application from a document type definition; and

ii) associating one or more javascript files, cascading style sheets, or behavior scriplet files with the meta-language application.

3 A method as in claim 2, wherein step c) further includes the step of:

i) enabling the remote server to function as a Web-server.

4 A method as in claim 3, wherein step c) further includes the step of:

i) updating the meta-language application by a third party developer by accessing the Web-server and providing custom versions of the files comprising the meta-language application.

5 A method for providing a Web-based user interface for a digital imaging device, comprising the steps of:

a) defining a markup language in a meta-language to provide a document type definition (DTD) file;

b) writing a graphical-user-interface (GUI) application based on the markup language;

c) writing a camera meta-language parser capable of interpreting and displaying applications that are based on the markup language;

d) installing the GUI application and the meta-language parser in the digital imaging device;

e) updating the GUI application;

f) posting the updated GUI application on a Web server; and

g) downloading the updated GUI application from the Web server into the digital imaging device to thereby change the user interface of the camera.

6 A method as in claim 5, wherein step e) further includes the steps of:

i) updating the GUI application by updating the DTD file.

7 A method as in claim 5, wherein step e) further includes the steps of:

i) associating the DTD file with a style sheet; and

ii) updating the GUI by updating the style sheet.

8 A method as in claim 5, further including the steps of:

a) writing a second graphical-user-interface application based on the markup language;

b) posting the second graphical-user-interface application on a Web server; and

c) downloading the second graphical-user-interface application from the Web server into the digital imaging device to thereby change the user interface of the camera.

9 A system for implementing a cross-platform graphical-user-interface (GUI) for a digital imaging device, comprising:

a memory containing a GUI application written in a meta-language, and a meta-language parser;

a processor coupled to the memory for accessing the GUI application and the meta-language parser, wherein when the processor executes the meta-language parser, the meta-language parser interprets the GUI application and displays a GUI on a display device;

means for downloading updates to the GUI application from a remote server,

whereby the updates to the GUI application are stored in the memory and interpreted by the meta-language parser to thereby change the GUI of the digital imaging device.

10 A system as in claim 9 wherein the remote server is a Web server.

11 A system as in claim 10 wherein the digital imaging device includes the display device.

12 A system as in claim 11 wherein the digital imaging device is a first digital imaging device and the processor is a first processor, the system further comprising;

a second digital imaging device having a second processor having an architecture different from the first processor, the second digital imaging device including,

the meta-language parser, and

means for downloading the updated GUI to provide the second digital imaging device with a GUI wherein the updated GUI does not have to be compiled for the first or second processors.

13 A computer-readable medium containing program instructions for providing a digital imaging device having a display with a Web-based graphical user interface (GUI), the program instructions for:

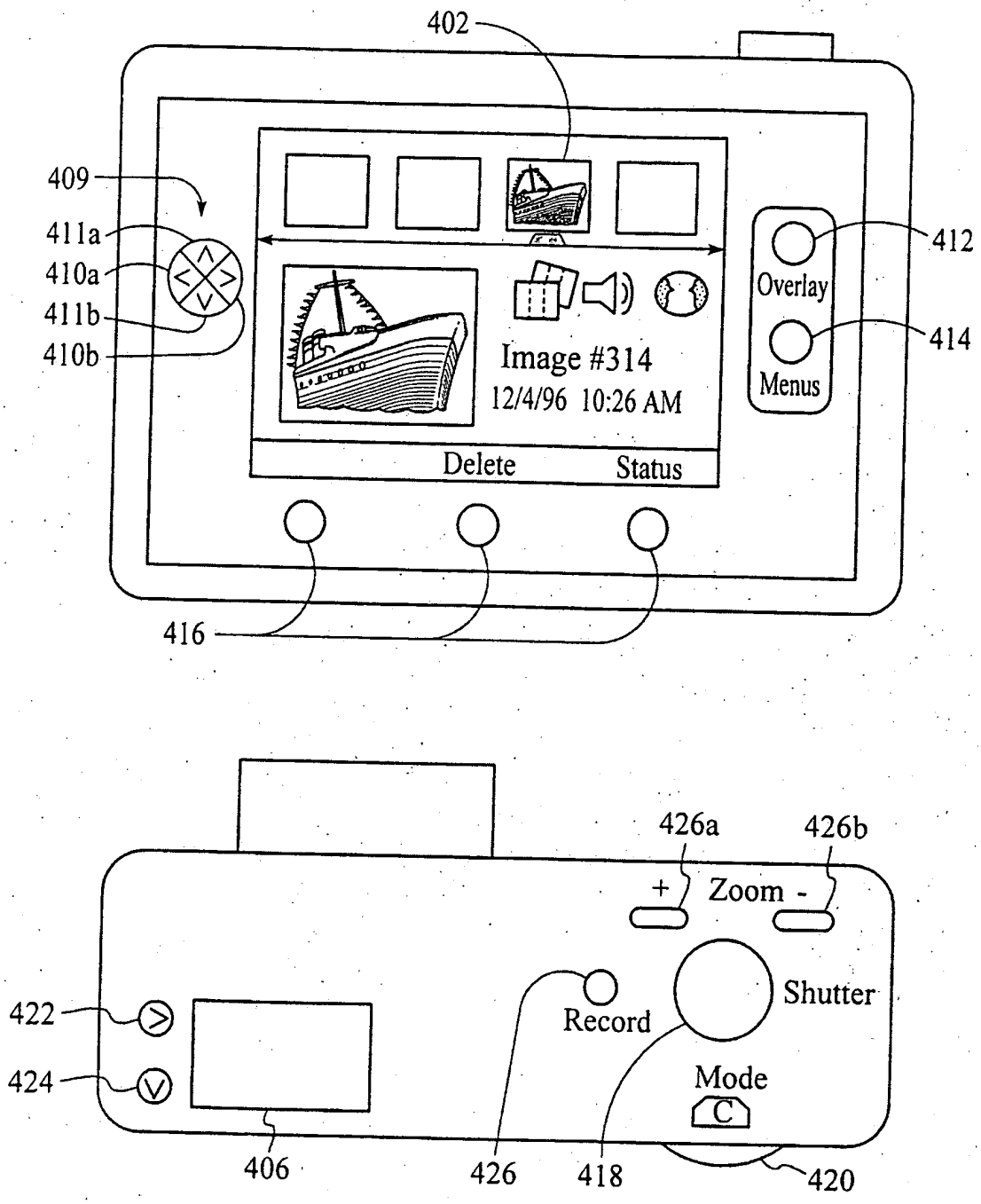
a) providing the digital imaging device with a meta-language application defining the GUI;

b) providing the digital imaging device with a meta-language parser for interpreting and displaying the meta-language application on the display; and

c) downloading an updated version of the meta-language application into the digital imaging device to thereby change the GUI of the camera.

14 A computer-readable medium as in claim 13 wherein program instructions c) further include instructions for:

i) accessing a remote server to downloading the updated version of the meta-language application.



110

FIG. 1

(PRIOR ART)

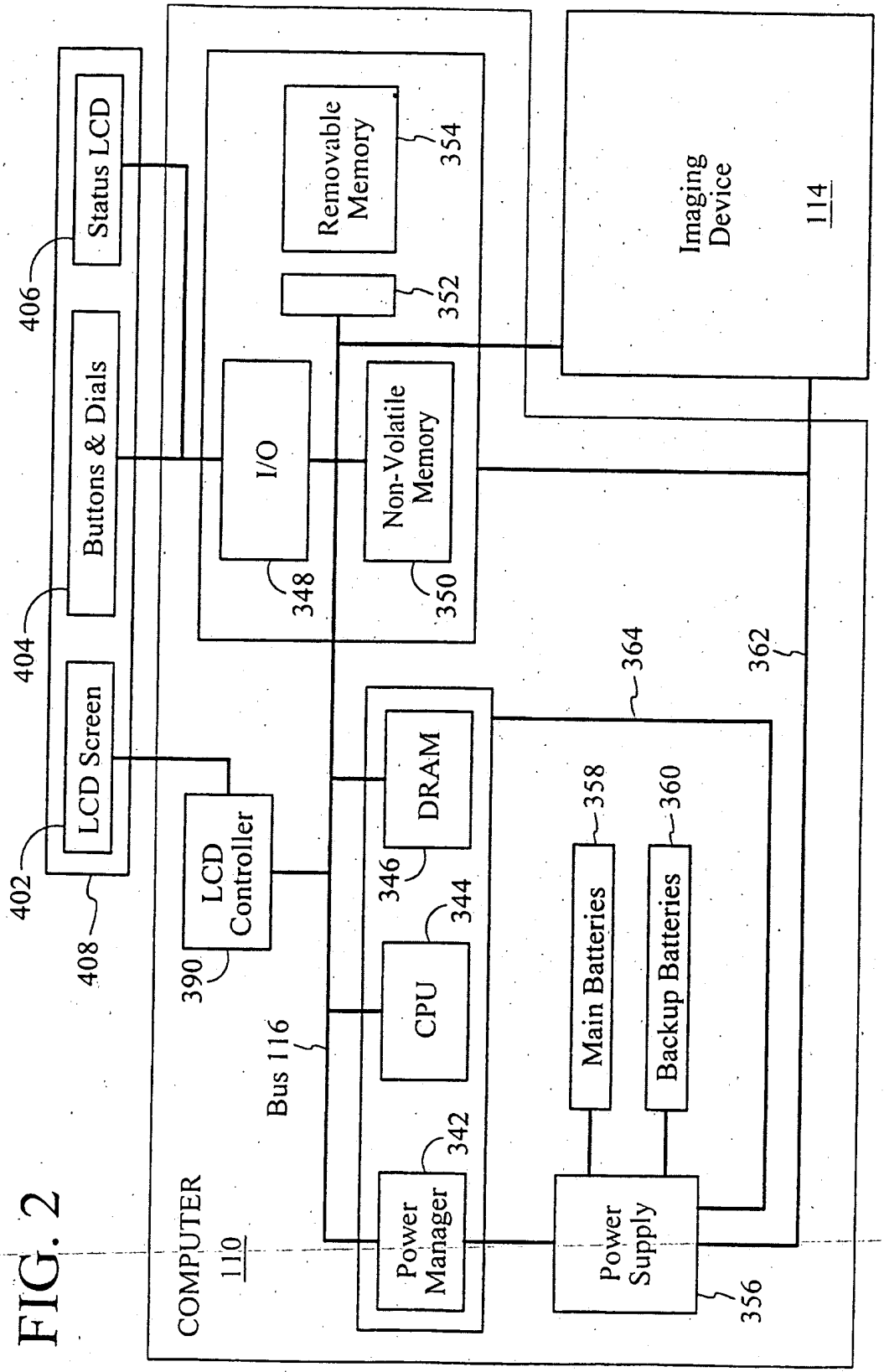

COMPUTER
110
Bus 116
Power Manager
342
CPU
344
DRAM
346
LCD Controller
390
LCD Screen
402
408
Buttons & Dials
404
Status LCD
406
I/O
348
Non-Volatile Memory
350
Removable Memory
354
352
Imaging Device
114
362
364
Main Batteries
358
Backup Batteries
360
Power Supply
356

FIG. 2

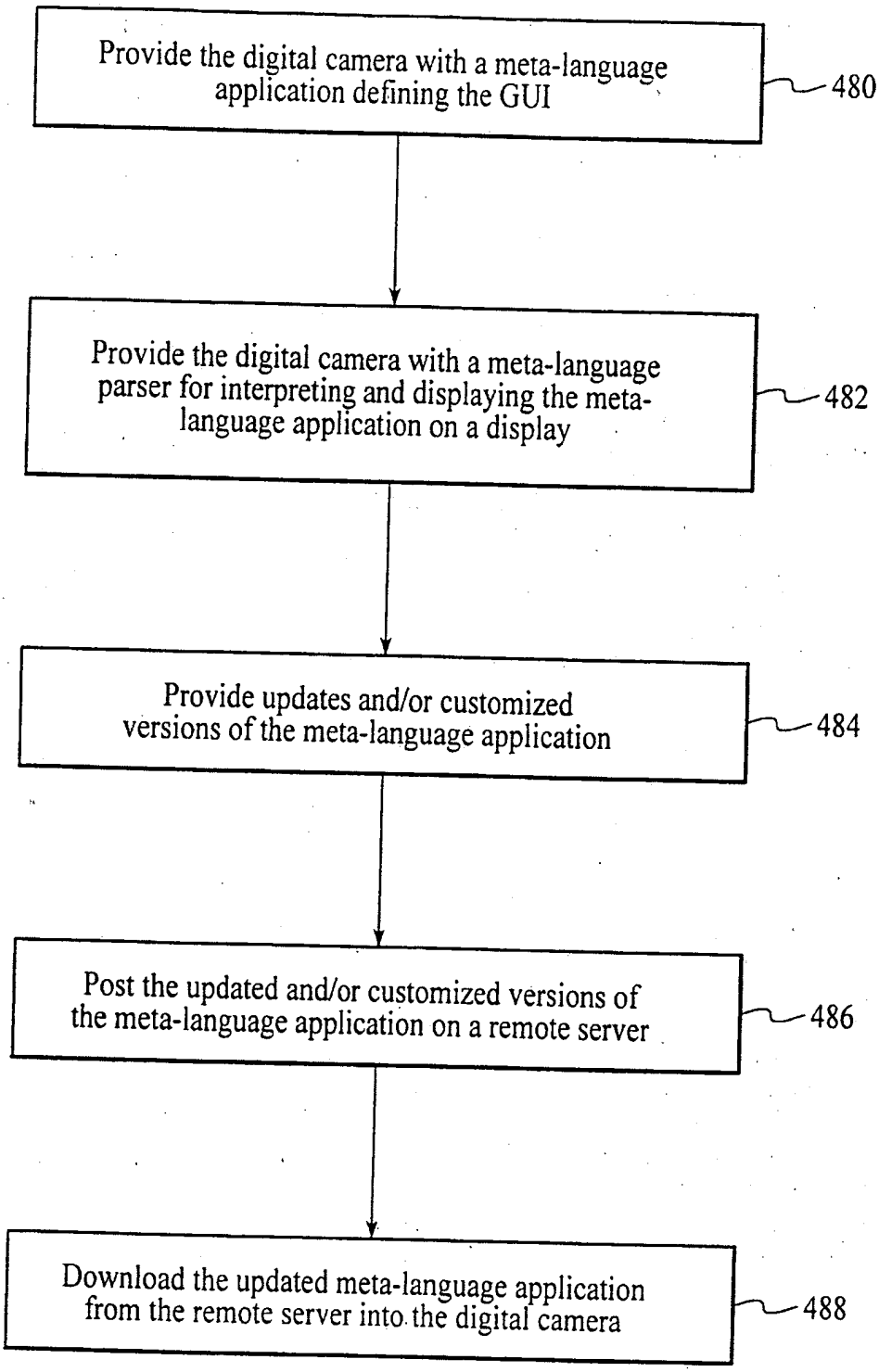

Provide the digital camera with a meta-language application defining the GUI
480
Provide the digital camera with a meta-language parser for interpreting and displaying the meta-language application on a display
482
Provide updates and/or customized versions of the meta-language application
484
Post the updated and/or customized versions of the meta-language application on a remote server
486
Download the updated meta-language application from the remote server into the digital camera
488

FIG. 3

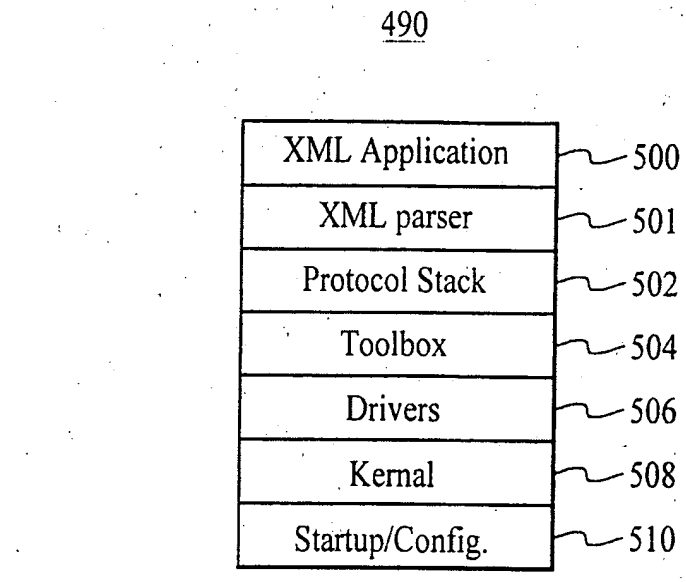

490
XML Application
500
XML parser
501
Protocol Stack
502
Toolbox
504
Drivers
506
Kernal
508
Startup/Config.
510
350

FIG. 4

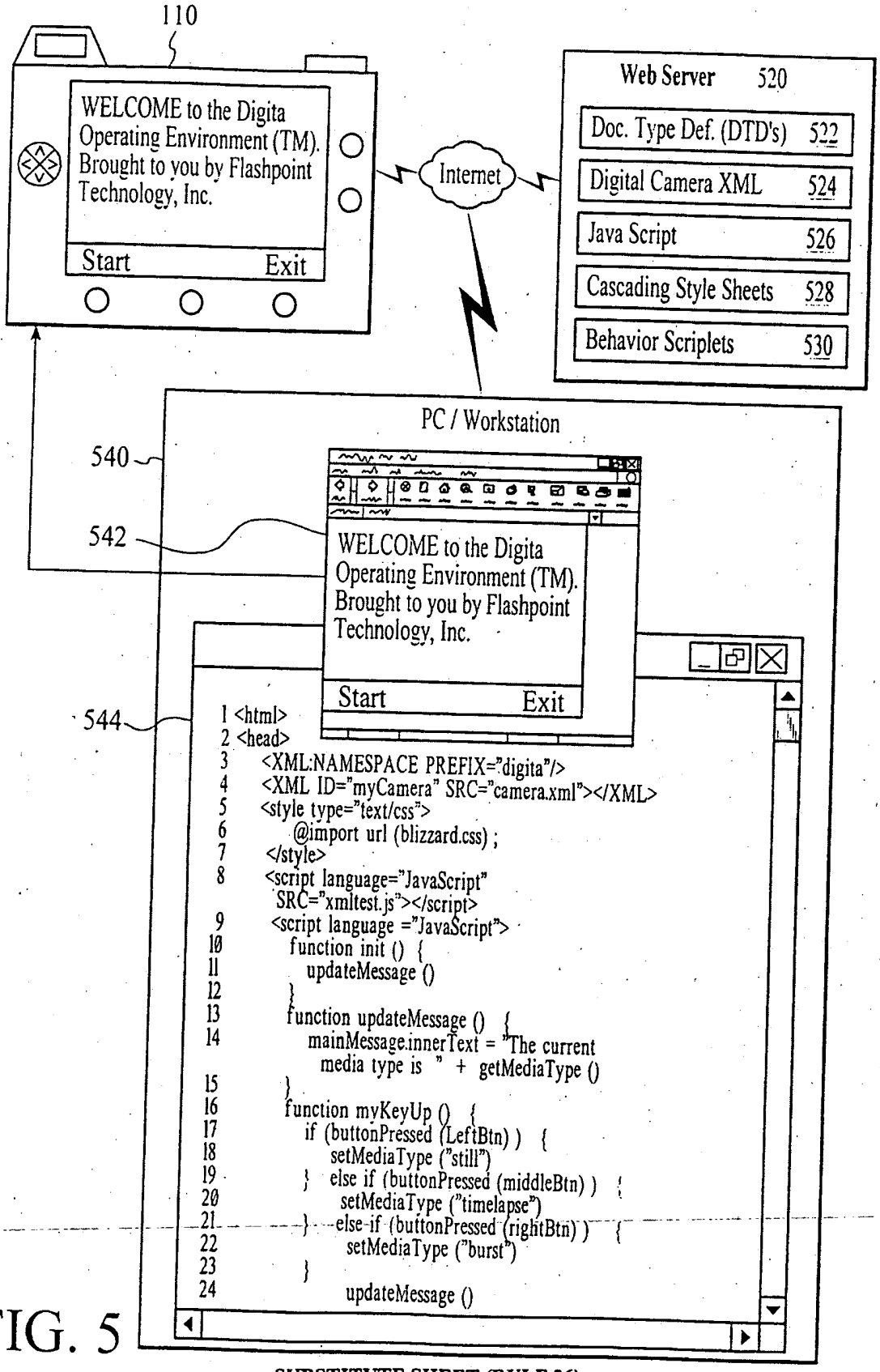

110
WELCOME to the Digita Operating Environment (TM). Brought to you by Flashpoint Technology, Inc.
Start
Exit
Internet
Web Server 520
Doc. Type Def. (DTD's) 522
Digital Camera XML 524
Java Script 526
Cascading Style Sheets 528
Behavior Scriplets 530
PC / Workstation
540
542
544
WELCOME to the Digita Operating Environment (TM). Brought to you by Flashpoint Technology, Inc.
Start
Exit
1 <html>
2 <head>
3 <XML:NAMESPACE PREFIX="digita"/>
4 <XML ID="myCamera" SRC="camera.xml"></XML>
5 <style type="text/css">
6 @import url (blizzard.css) ;
7 </style>
8 <script language="JavaScript" SRC="xmltest.js"></script>
9 <script language ="JavaScript">
10 function init () {
11 updateMessage ()
12 }
13 function updateMessage () {
14 mainMessage.innerText = "The current media type is " + getMediaType ()
15 }
16 function myKeyUp () {
17 if (buttonPressed (LeftBtn)) {
18 setMediaType ("still")
19 } else if (buttonPressed (middleBtn)) {
20 setMediaType ("timelapse")
21 } else if (buttonPressed (rightBtn)) {
22 setMediaType ("burst")
23 }
24 updateMessage ()

FIG. 5

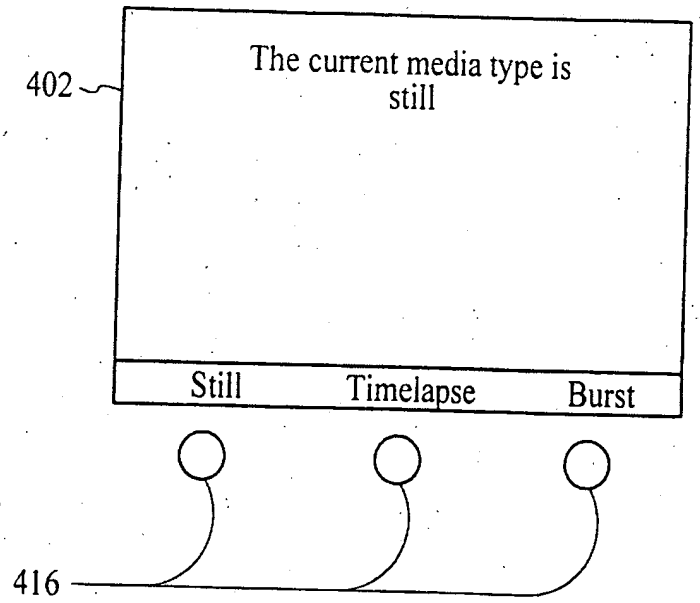

402
The current media type is still
Still
Timelapse
Burst
416

FIG. 6A

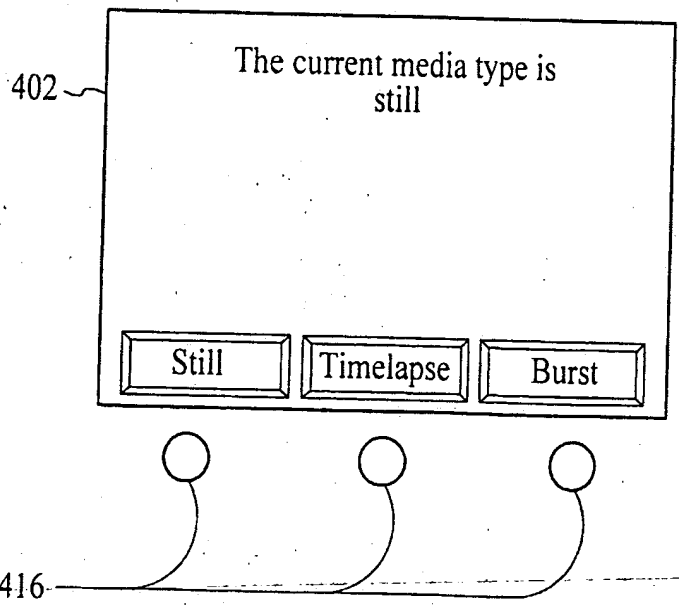

402
The current media type is still
Still
Timelapse
Burst
416

FIG. 6B

```
:
:
digita\:sk1 {
        keyCodeRef: 49;
        position: absolute;
        top: -4px; left: 2px;
        width: 93px; height: 24px;
        color: white;
        font-family: "ITC Officina Sans Book";
        font-size: 24;
        font-weight: bold;
        text-align: left;
}
:
:
```

FIG. 7A

```
:
:
digita\:sk1 {
        keyCodeRef: 49;
        position: absolute;
        top: 0px; left: 2px;
        width: 93px; height: 24px;
        color: rbg(251,252,208);
        font-family: "Arial"; font-size: 18;
        font-weight: bold;
        text-align: center;
        border-style: solid;
        border-width: 2px;
        border-left-color: rgb(121, 118, 197);
        border-top-color: rgb(121, 118, 197);
        border-right-color: rgb(24, 24, 48);
        border-bottom-color: rgb(24, 24, 48);
}
:
:
```

FIG. 7B

```
1 <html>
2 <head>
3     <XML:NAMESPACE PREFIX="digita"/>
4     <XML ID="myCamera" SRC="camera.xml"></XML>
5     <style type="text/css">@import url (xmltest.css);</style>
6     <script language="JavaScript" SRC="xmltest.js"></script>
7     <script language="JavaScript">
8               function unit () {
9                       updateMessage ()
10            }
11            function updateMessage ()  {
12            mainMessage.innerText = "The current media type is"
                                                          + getMediaType ()
13            }
14            function myKeyUp ()  {
15                      if (buttonPressed (LeftBtn) )  { setMediaType ("still")
16                      }  else if (buttonPressed (middleBtn) )  {
17                              setMediaType ("timelapse")
18                      }  else if (buttonPressed (rightBtn) )  {
19                              setMediaType ("burst")
20                      }
21                              updateMessage ()
22            }
23      </script>
24 </head>
```

FIG. 8

```
1  <body scroll="no" onLoad="init()" bgcolor="black">
2   <digita:screen onKeyUp="myKeyUp()">
3       <digita:appName>Demo</digita:appName>
4       <digita:scrnName>Media Type</digita:scrnName>
5       <digita:page>
6               <digita:message id="mainMessage"> </digita:message>
7       </digita:page>
8       <digita:cmdbar>
9               <digita:sk1 id="leftBtn">Still</digita:sk1>
10                      <digita:sk2 id="middleBtn">Timelapse</digita:sk2>
11                      <digita:sk3 id="rightBtn">Burst</digita:sk3>
12      </digita:cmdbar>
13    </digita:screen>
14  </body>
15  </html>
```

FIG. 9

```
1  <!ELEMENT camera (mediatype, imagecount, quality)>
2
3   <!ELEMENT mediatype (#PCDATA)>
4   <!ELEMENT imagecount (#PCDATA)>
5   <!ELEMENT quality (#PCDATA)>
6
7   <!ATTLIST camera mode (capture|review|play|connect) "capture">
8
9   <!ENTITY lt "&#38;#60;">
10  <!ENTITY gt "&#62;">
11  <!ENTITY amp "&#38;#38;">
12  <!ENTITY apos "&#90;">
13  <!ENTITY quote "&#34;">
```

FIG. 10

PCT/US00/08325

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 3/00
US CL : 345/333, 335

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 345/333, 334, 335, 326, 327; 348/239, 333

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
BRS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 5,903,309 A (ANDERSON) 11 May 1999 (11.05.1999), figure 4, abstract	1-14
Y, P	US 5,982,362 A (CRATER et al) 09 November 1999 (09.11.1999), abstract; col. 2, lines 47-52; col. 3, lines 34-42; col. 6, lines 4-24	1-14

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
	25 JUL 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Raymond Bayerl James R. Matthews Telephone No. (703) 305-9789

Form PCT/ISA/210 (second sheet) (July 1998)